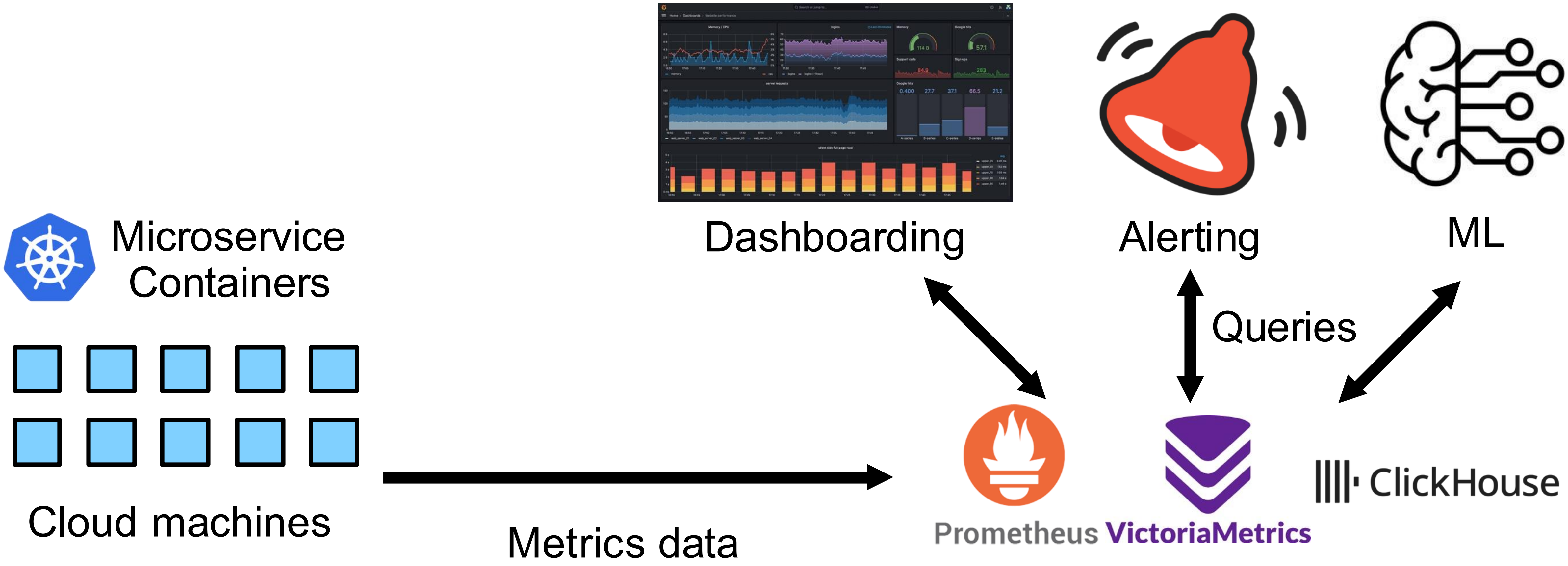
Approximation-First Timeseries Query At Scale

Zeying Zhu, Jonathan Chamberlain†, Kenny Wu,
David Starobinski†, Zaoxing Liu
University of Maryland, †Boston University

UNIVERSITY OF MARYLAND

BOSTON UNIVERSITY

Need for Cloud and Network Observability

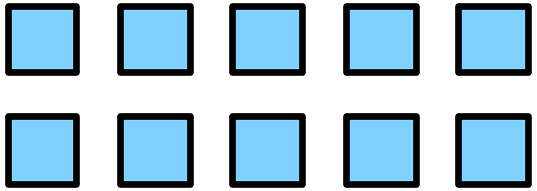


Example: Monitoring Cloud Machines

CPU Usage /Timestamps	5:00:00	5:00:01	5:00:02	5:00:03	…
Host 1	10%	35%	15%	20%	
Host 2	20%	15%	19%	20%	
Host 3	15%	75%	80%	85%	
Host 4	90%	80%	95%	75%	
…					

Queries

Microservice Containers



Cloud machines

Metrics data

Prometheus

Example *Rule* Queries for Cloud Resource Scaling

CPU Usage /Timestamps	5:00:00	…	5:01:00	…	5:02:00	…	5:03:00	…	5:04:00	…	5:05:00	…	5:06:00	…	5:07:00	…
Host 1	10%	…	35%	…	15%	…	20%	…	15%	…	20%	…	50%	…	15%	…
Host 2	20%	…	15%	…	19%	…	20%	…	19%	…	10%	…	40%	…	19%	…
Host 3	15%	…	75%	…	80%	…	85%	…	81%	…	80%	…	83%	…	80%	…
Host 4	90%	…	80%	…	95%	…	75%	…	85%	…	95%	…	80%	…	95%	…
…														…		…

Sliding Window Queries with overlaps between windows

- *Rule* Queries: Periodically (every 1 min) querying 0.95-quantile of CPU usage of last 5 min, for each host.
  - quantile_over_time(0.95, cpu_usage[5m])

Bottlenecks in Window-based Queries

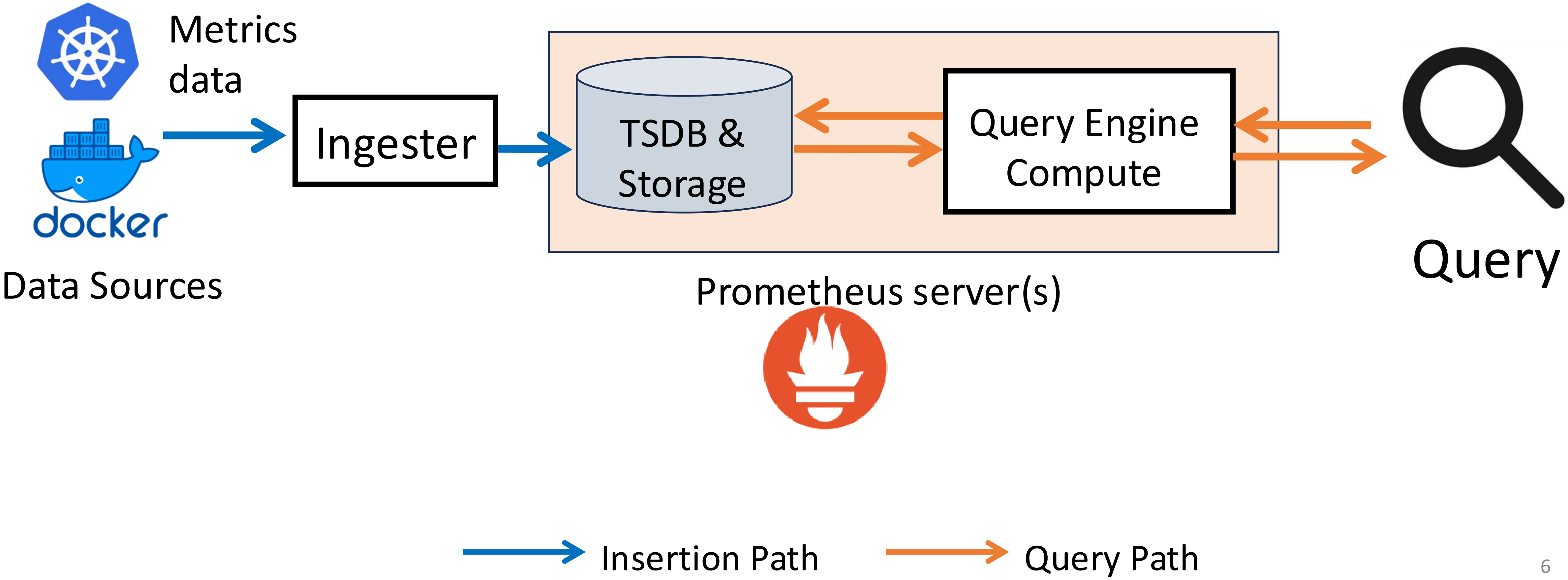
Table 2: CPU hotspots of evaluating a quantile rule query in Prometheus and VictoriaMetrics.

Func/Call Stack	CPU Time		Description
	Prometheus	VictoriaMetrics	
Data Scanning	**41%**	**80.2%**	Fetch data from storage
Query computation	**27.6%**	**11.7%**	Aggregation queries in rule
Go Garbage Collector	24.7%	4.3%	Golang garbage collector
mcall	4.5%	0.8%	Golang runtime scheduling

Major Bottlenecks due to window overlaps:

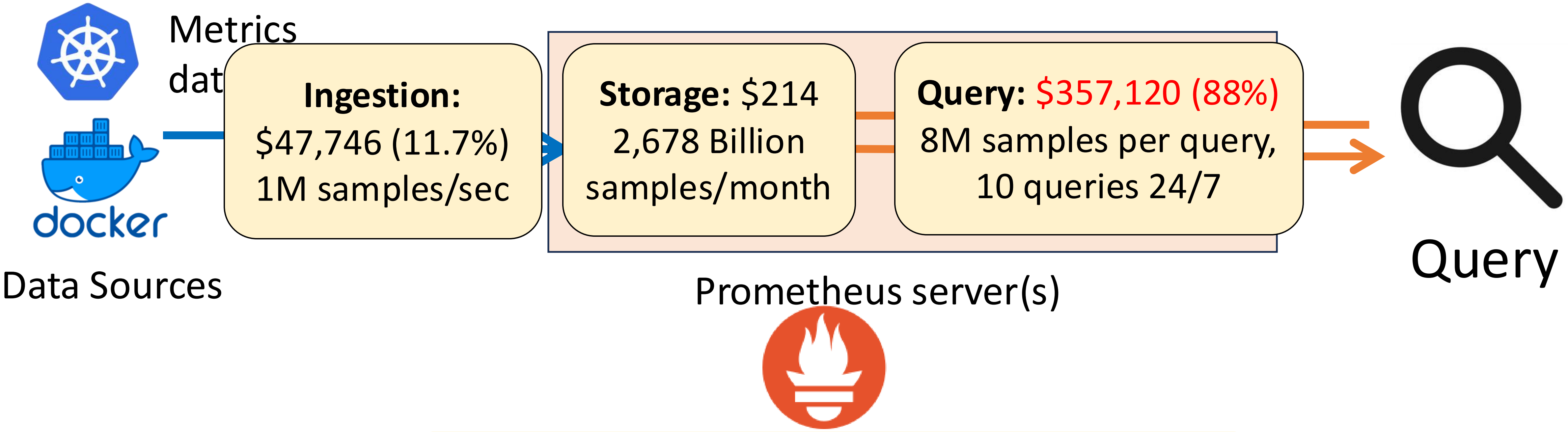
- Repeated data scanning from storage.
- Repeated and heavy query computation.

Prometheus is a Popular Observability Platform



Cloud Monitoring Costs Example

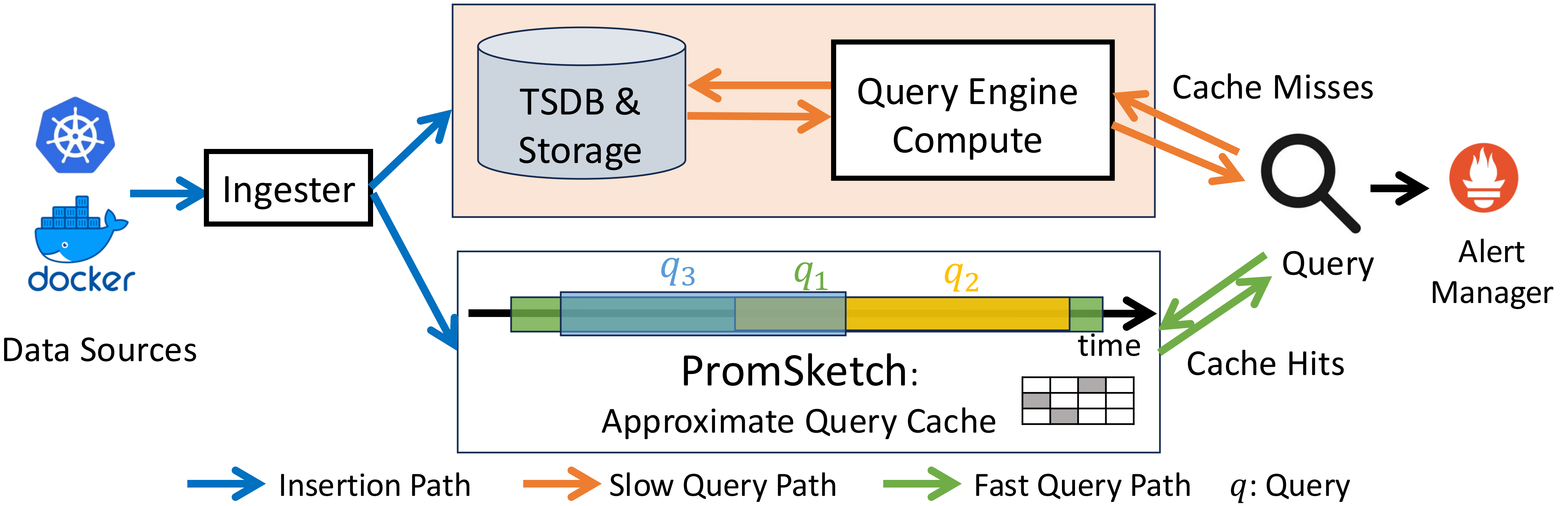
Setup: monitoring 1000-node Kubernetes cluster, each node having 1000 metrics.
AWS Prometheus Pricing: Charge users based on the number of data samples processed.



Query costs are very high in cloud monitoring.

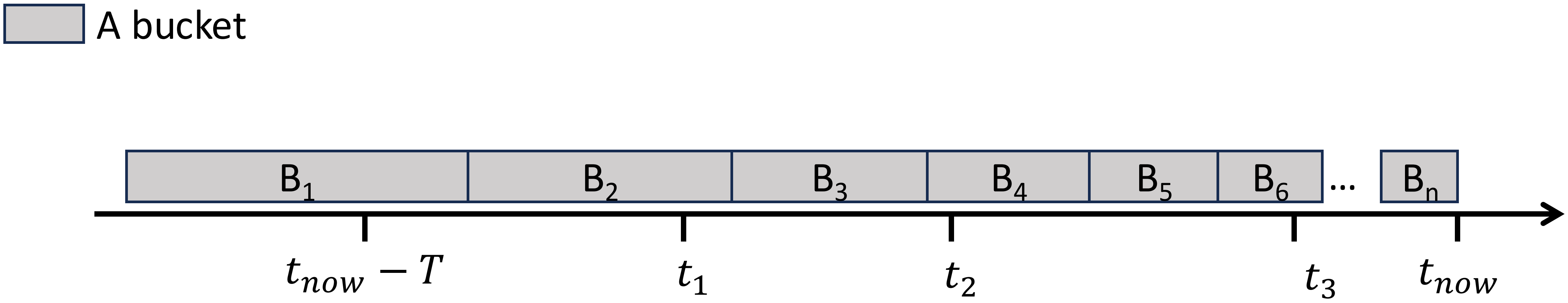
Insertion Path Query Path

PromSketch as an Intermediate Cache



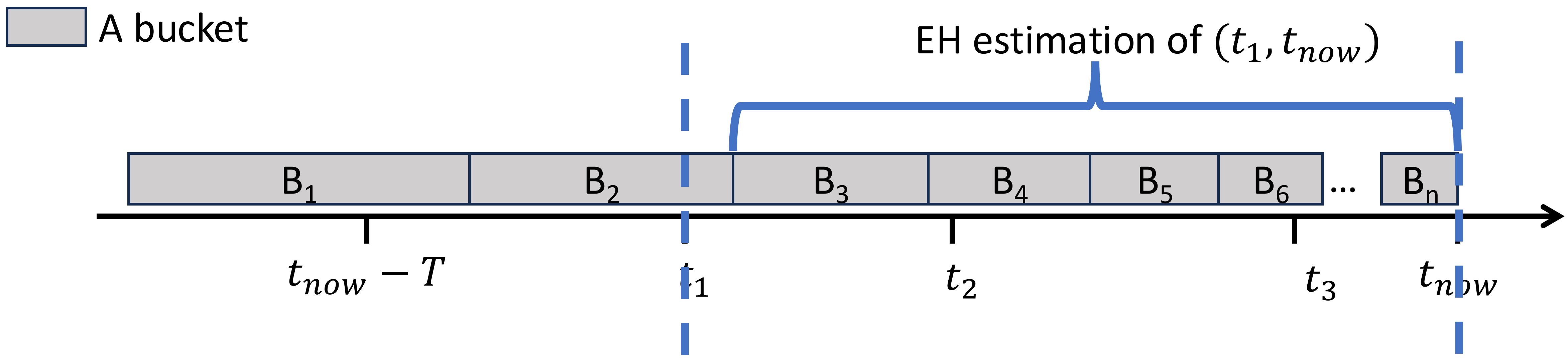
Key Idea: Combining arbitrary sub-window query frameworks and compact sketches as an intermediate query cache, computing overlapping windows once.

Exponential Histogram (EH) Framework



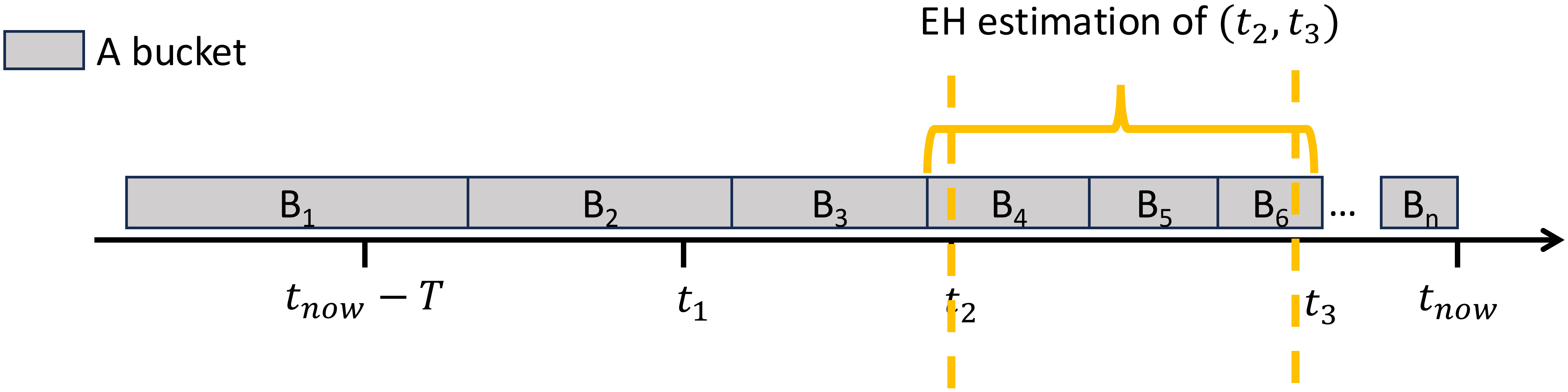
- EH supports querying arbitrary sub-windows cached by most recent T.

Querying Sub-windows by Merging Buckets



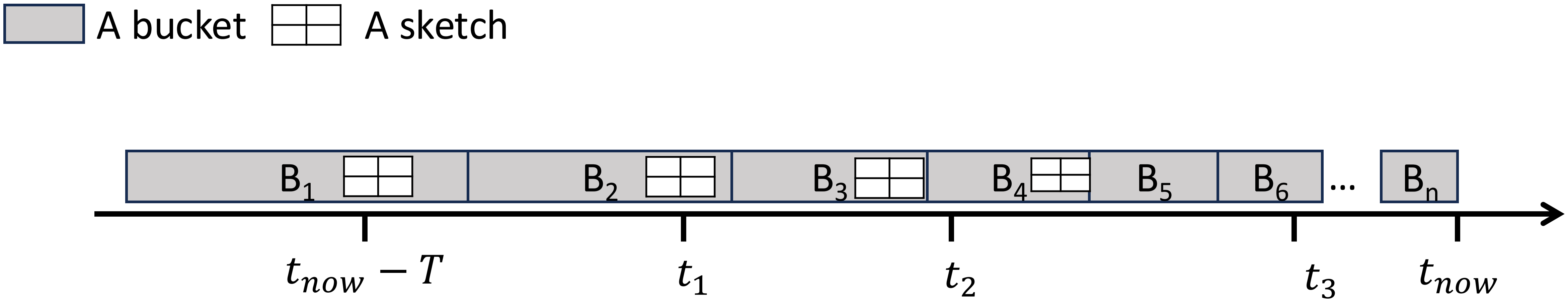
- EH supports querying arbitrary sub-windows cached by most recent T.
  - Combining buckets closest to the query time range.

Querying Sub-windows by Merging Buckets



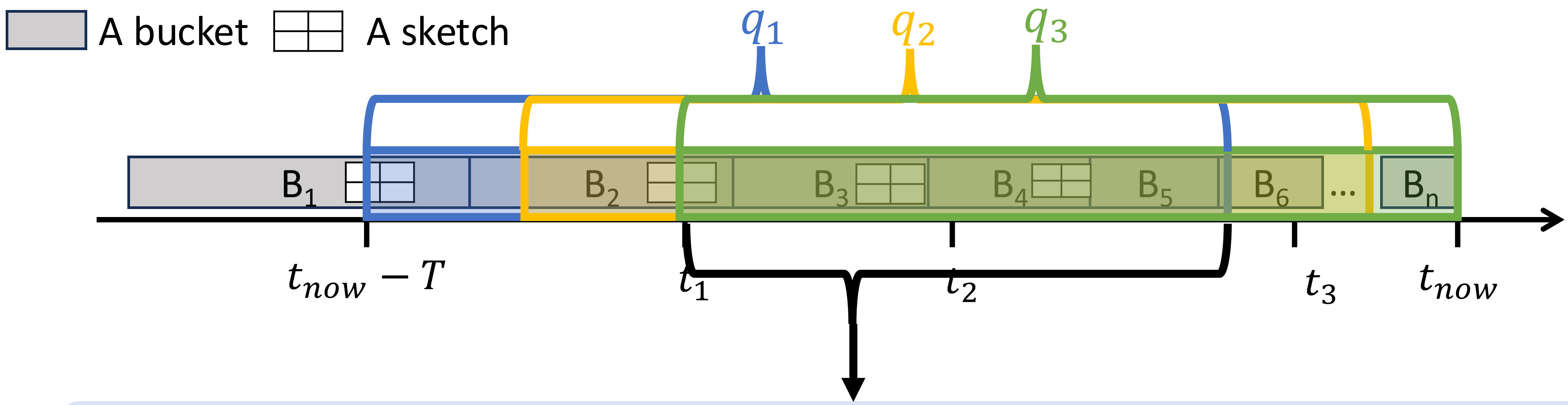
- EH supports querying arbitrary sub-windows cached by most recent T.
  - Combining buckets closest to the query time range.

Provable Compact Sketches as Buckets



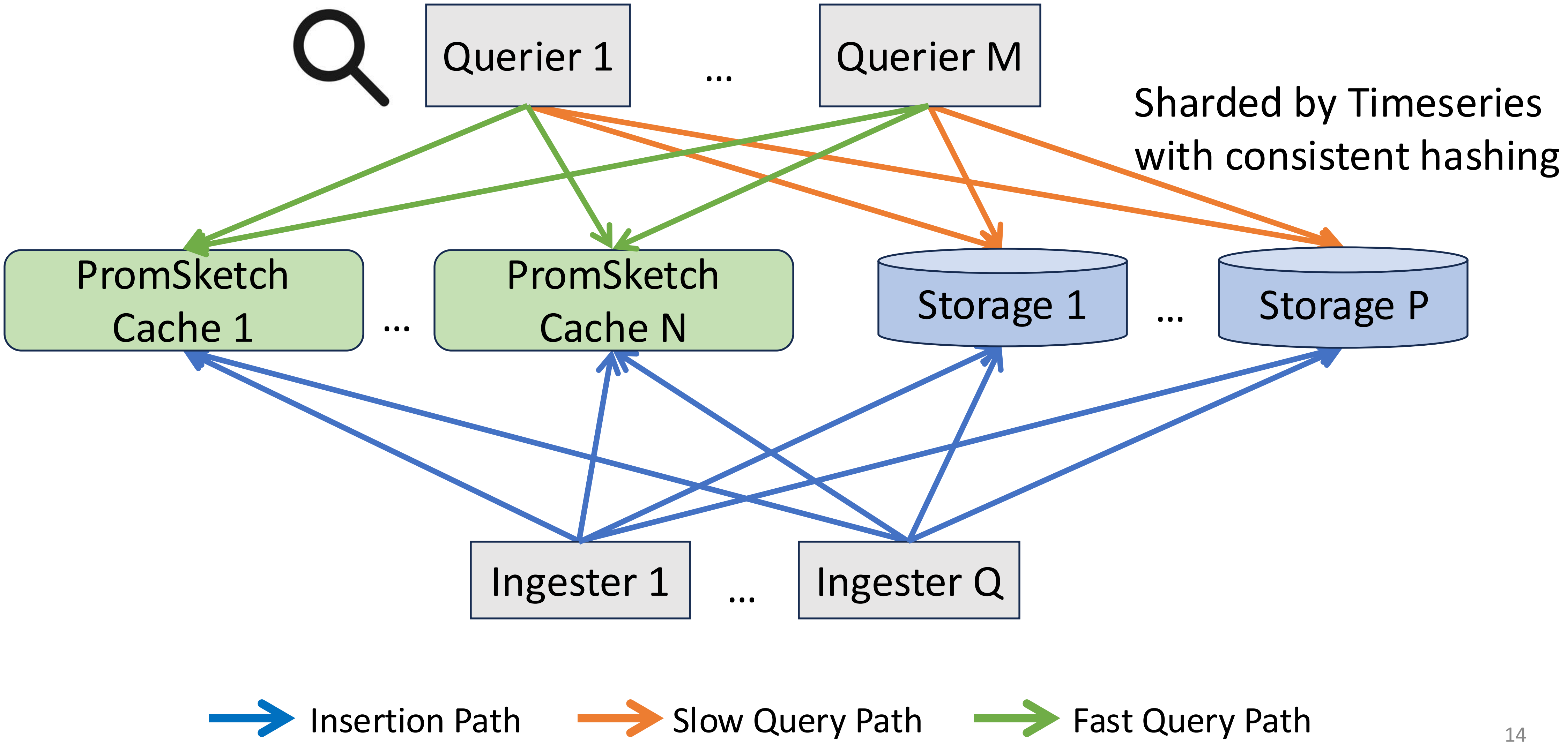
- KLL Sketch for Quantiles, Min, and Max.
- Universal Sketching for a wide range of functions, e.g., cardinality, L2 norm, entropy, top-K finding.

Precomputed Intermediate Cache Eliminates Query Overhead



- Overlapping windows are only computed once and cached in memory.

PromSketch Supports Cluster Version



Evaluation

- Testbed
  - Ubuntu servers with a 32-core CPU, 384GB DDR4 memory, and 1TB Seagate HD
- Baselines
  - Prometheus (single-machine)
  - Single-machine and cluster version VictoriaMetrics
  - Single-machine and distributed PromSketch
- Workloads
  - Google Cluster Dataset 2019 (Google), CAIDA NYU 2018 and CAIDA NYU 2019
  - Dynamic dataset with Zipf, uniform, normal distribution data samples
  - 10 million records

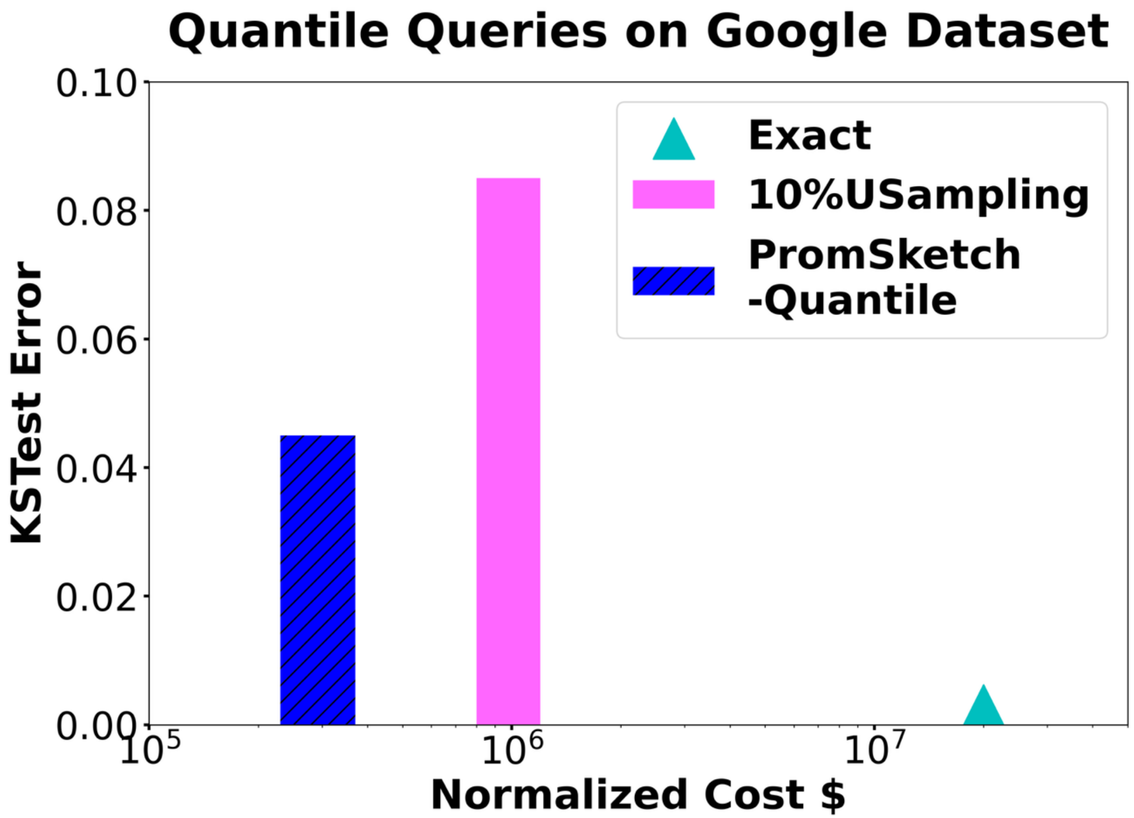
End-to-End Total Concurrent Query Latency

Total concurrent rule query latency on 10K-, 100K-, and 1M-sample windows

Metrics	Prometheus	Prometheus-PromSketch	VictoriaMetrics (VM)	VM-PromSketch
0.9-Quantile	5005 sec	28 sec **(181x ↓)**	96.1 sec	3.2 sec **(30x ↓)**
0.9-Quantile &Max &Average &Distinct	13177 sec	88.6 sec **(154x ↓)**	590 sec	9 sec **(65x ↓)**

- Up to 231x concurrent query latency reduction compared to Prometheus.
- Up to 158x compared to single-machine VictoriaMetrics.

Accuracy-Operational Cost Tradeoffs



- Achieve better cost-accuracy tradeoffs compared to uniform sampling.
- Achieve 5% mean errors with 5x~75x smaller cost compared to exact computation.

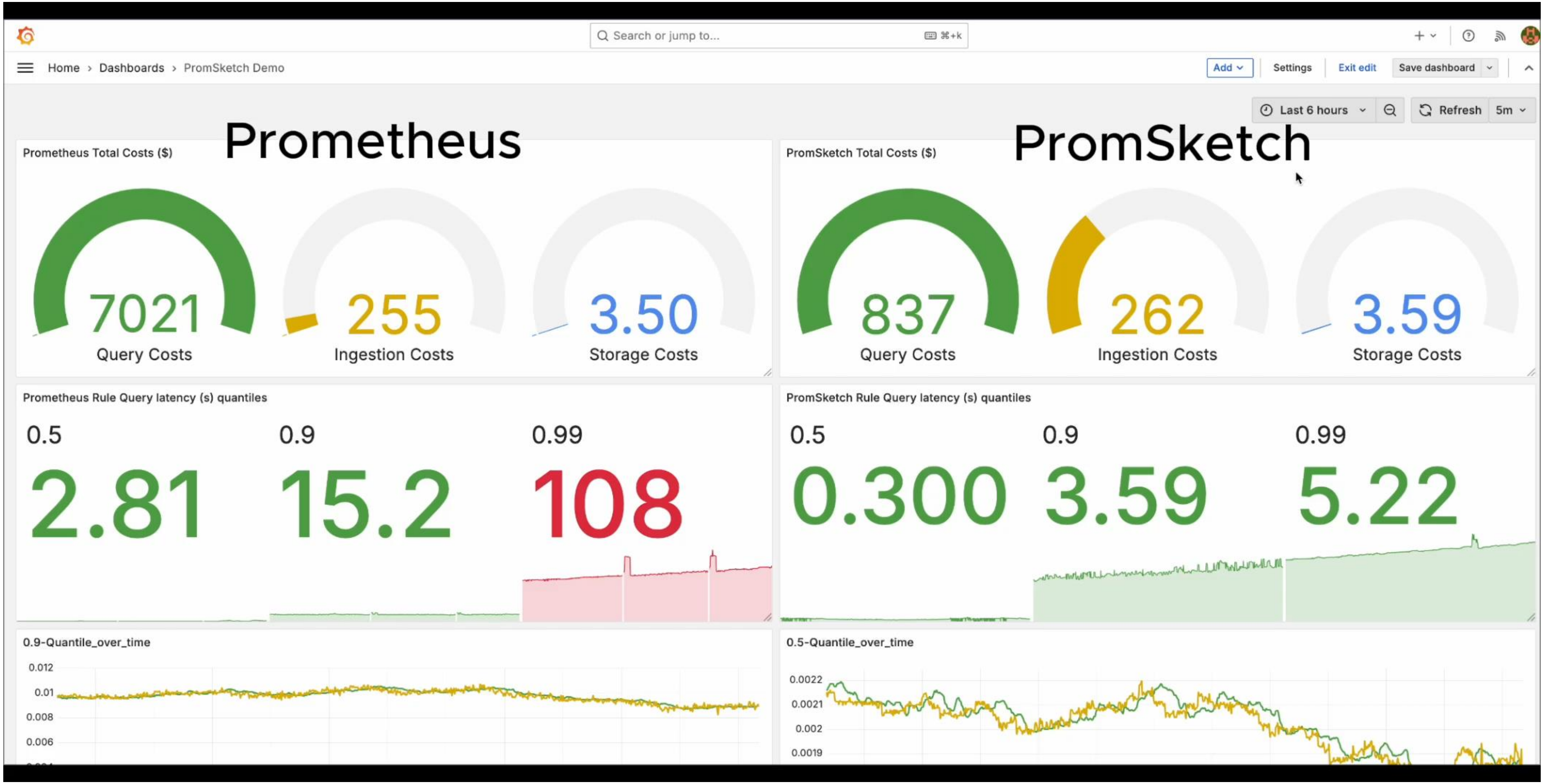
PromSketch Saves Monitoring Costs

Setup: monitoring 1000-node Kubernetes cluster, each node having 1000 metrics.

Resources	Data Volume	AWS Prometheus Costs/Month	Prometheus-PromSketch
Storage	2,678 Billion samples/month	$214	$214
Data Ingestion	1M samples/sec	$47,746	$47,746
Query Processing	8M samples per query, 10 queries running 24/7	**$357,120**	**$267.8 (1,334x ↓)**
Total Costs		$405,080	$48,227.8

- Reducing query costs by up to three orders of magnitude.

It’s Demo Time!



PromSketch Conclusions

- Cloud timeseries monitoring systems, such as Prometheus, incur significant operational costs and high query latency.
- We design PromSketch, an approximation-first query processing framework that leverages sub-window query frameworks and sketch-based precomputation as intermediate caches.
- Integrated with Prometheus & VictoriaMetrics, open-sourced at https://github.com/Froot-NetSys/promsketch.

Thank you!